Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им.Н.Э. Баумана)

Факультет «Робототехника и комплексная автоматизация» Кафедра «Системы автоматизированного проектирования»

# Отчет

**по лабораторной работе №7,8 по дисциплине**

# «Программное обеспечение САПР»

# Вариант A5

Выполнил: студент группы РК6-32 Журавлев Н.В.

Проверил: старший преподаватель Волосатова Т.М.

Москва

2020

## Техническое задание

## Постановка задачи

## 

**Введение**

Настоящий документ определяет техническое задание на разработку программы транслятора двоичных наборов, которые состоят из нулей или только из единиц (далее по тексту - программа L5).

## Основания для разработки

Программа разрабатывается в рамках лабораторной работы по курсу "Лингвистическое обеспечение САПР" для практического изучения этапов лексического и синтаксического анализа в процедурах трансляции формальных языков.

## Назначение разработки

Программа предназначается для реализации лексического и грамматического разбора, заключающегося в распознавании необходимых десятичных наборов чисел, в системах и модулях обработки десятичных последовательностей.

## Требования к программе

1. ***Требования к функциональным характеристикам***
   1. Программа должна в интерактивном режиме распознавать записей двоичных наборов, которые соответствуют формальному языку L={0n1n | n>0}.
   2. Двоичные наборы должны передаваться строками стандартного.
   3. Программа обнаруживает соответствие или несоответствие бинарной последовательности заданным правилам, и ошибки ввода
   4. При нарушении ввода или несоответствии последовательности заданным правилам, программа должна отображать диагностическое сообщение “syntax error” в потоке стандартного вывода.
   5. Результат распознавания программой бинарной последовательности, соответствующей заданным правилам, должен сопровождаться отображением в потоке стандартного вывода информационным сообщением "Correct".

## Требования к надежности

Программа не должна иметь каких-либо ограничений по числу символов в анализируемой бинарной последовательности, кроме внутренних ограничений инструментальных средств, использованных для ее реализации.

## Требования к составу и параметрам технических средств

Программа должна быть разработана исходя из возможности реализации на стандартном составе технических средств компьютеров любой архитектуры, после соответствующей трансляции исходного кода.

## Условия эксплуатации

* 1. Программа должна быть ориентирована на эксплуатацию в среде OS UNIX.
  2. Программа должна быть реализована в виде выполняемого файла с именем y5, по которому она должна вызываться средствами любого командного процессора OS UNIX.
  3. Программа должна эксплуатироваться в интерактивном режиме, читая строки из потока стандартного ввода и отображая результаты их обработки в потоке стандартного вывода.

## Требования к информационной и программной совместимости

* 1. Чтобы обеспечить выполнение требуемых технических характеристик, программа должна реализовывать синтаксический анализ любой входной строки скобочного выражения из потока стандартного ввода.
  2. Синтаксический анализатор программы должен обеспечивать грамматический разбор двоичных наборов с целью установить соответствие или несоответствие содержащей их строки потока стандартного ввода заданным правилам.
  3. Для выполнения грамматического разбора синтаксический анализатор программы должен реализовывать однозначную контекстно-свободную грамматику, которая ориентирована на обработку строки заданной бинарной последовательности из потока стандартного ввода, и далее по тексту называется грамматикой БП.
  4. Грамматику БП синтаксического анализатора программы должны составлять следующие элементы:
* терминальные символы (терминалы), соответствующие структурным единицам (лексемам) входного выражения;
* начальный нетерминальный символ (начальный нетерминал), к которому приводится входное выражение;
* нетерминальные символы (нетерминалы), обозначающие допустимые варианты комбинации лексем во входном скобочном выражении;
* система продукций (правил вывода), обеспечивающая грамматический разбор входного скобочного выражения.
  1. Терминальные символы грамматики БП синтаксического анализатора программы должны представляться лексемами, специфицированными следующими литералами:

## ‘ZERO’,’ONE’,’\n’

которые обозначают коды ASCII нуля, единицы и перевода строки.

* 1. Начальный нетерминал грамматики БП синтаксического анализатора программы должен обозначаться именем input. Он должен выводиться из любой корректной или ошибочной последовательности.
  2. Для разработки синтаксического анализатора программы , необходимо использовать генератор синтаксических анализаторов (далее по тексту - YACC) из состава OS UNIX, инструментальные средства которого ориентированы на обработку файла спецификаций (далее по тексту, Yacc-файл) проектируемого синтаксического анализатора.
  3. При разработке синтаксического анализатора программы необходимо составить Yacc-файл, отражающий специфику грамматического разбора скобочного выражения, и сохранить его под именем l5.y в выбранном доступном рабочем каталоге файловой системы OS UNIX.
  4. Проектируемый Yacc-файл l5.y должен состоять из 3-х секций: деклараций, правил и функций. Разделителем секций должны быть символические пары %%, расположенные в начальных позициях содержащих их строк Yacc-файла.
  5. Секция деклараций Yacc-файла l5.y должна включать:
     + cпецификацию блока внешних описаний, ограниченную директивами %{ и %}, в которой необходимо подключить библиотеку ввода/вывода stdio.h, а также необходимо явно объявить функции yyerror и yylex .
     + объявление с помощью директивы %token терминальных символов ZERO и ONE, задающих нуль и единицу соответственно.
  6. В секции правил Yacc-файла l5.y должны быть приведены описания продукций приведения нетерминалов грамматики БП.
  7. Продукции секции правил Yacc-файла l5.y, приведение нетерминалов которых необходимо сопровождать функциональной обработкой, должны содержать блоки действий. Блоки действий должны располагаться в правых частях продукций и ограничиваться парой

фигурных БП. Внутри блоков действий можно использовать любые конструкции и вызовы функций системы программирования C.

* 1. Продукции секции правил Yacc-файла l5.y, необходимые для приведения нетерминалов, должны быть специфицированы с помощью альтернативных правил. В каждом из них альтернативы свертки различных нетерминалов правой части должны быть объединены с помощью оператора ИЛИ, который обозначается символом вертикальной черты (|).
  2. Секция функций Yacc-файла l5.y должна содержать исходный код, оформленный по правилам системы программирования C, для функции yylex() которая должна иметь целочисленный код возврата.
  3. Для достижения целей лексического анализа, указанных в п. 5.17, исходный код функции yylex() должен предусматривать:
* побайтное чтение любой заданной строки входного скобочного выражения из потока стандартного ввода с помощью библиотечной функции getchar();
* распознавание символов во входной строке;
* возврат лексемы ONE, если была считана единица, и лексемы ZERO – если был считан нуль;
  1. Исходный код секции функций Yacc-файла l5.y образует исходный код программы , который должен формироваться генератором синтаксических анализаторов YACC в файле с предопределенным именем l5.c в текущем рабочем каталоге файловой системы OS UNIX. Выполняемый модуль программы должен быть создан по файлу исходного кода l5.c в выполняемом файле l5 средствами компилирующей системы программирования C.

## Стадии и этапы разработки

Процесс разработки программы целесообразно разделяться на следующие 3 стадии:

* составить Yacc-файл l5.y в выбранном рабочем каталоге файловой системы OS UNIX, используя любой текстовый редактор;
* получить исходный код синтаксического анализатора в файле l5.c текущего каталога файловой системы OS UNIX, обработав Yacc-файл l5.y командой yacc, следующим образом:

**$ yacc -d l5.y**

Сформировать выполняемый модуль в файле l5 текущего каталога файловой системы OS UNIX, компилируя исходный код синтаксического анализатора в файле y.tab.c следующей командой:

# $ cc -o l5 y.tab.c

Результаты разработки программы должны содержать описание грамматики и файл спецификаций для генератора синтаксических анализаторов YACC.

## Порядок контроля и приемки

1. Для проверки функционирования программы должны быть предложены контрольные примеры, предусматривающие стандартный ввод корректных и некорректных скобочных выражений с учетом вложенности и конкатенации блоков скобок различных типов.
2. Для приемки программы должен быть организован вызов выполняемого файла l5 в консольном режиме работы OS UNIX или режиме эмуляции терминала операционной среды X Window System.

**Результаты разработки**

# Описание грамматики

Терминальный алфавит: Σ = {'\n',ZERO, ONE} Нетерминальный алфавит: { one, zero };

Правила:

* 1. input -> {printf("Enter the line:\n");} – выводит на экран сообщение: "Enter the line: ";
  2. input -> input line – считывает строку;
  3. line -> '\n' {printf("Empty line\n");} – в случае, если на вход подана пустая строка, выводит диагностическое сообщение: "Empty line";
  4. line -> zero '\n' {printf ("Correct\n");} – в случае, если на вход подана бинарная последовательность, которая удовлетворяет указанному правилу, выводит диагностическое сообщение “Correct”;
  5. line -> one '\n' {printf ("Correct\n");} – в случае, если на вход подана бинарная последовательность, которая удовлетворяет указанному правилу, выводит диагностическое сообщение “Correct”;
  6. zero -> zero ZERO – свёртка двух символов “0”;
  7. zero -> ZERO – символ “0”
  8. one -> one ONE - свёртка двух символов “1”;
  9. one-> ONE – cимвол „1‟;

Начальный нетерминал: input

# Результаты синтаксического анализа

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 1111 00 1111 11a11 11 | CorrectCorrectCorrectCorrectCorrectsyntax errorCorrect |

# Содержимое файла спецификации c левосторонним выводом

%{

#include <stdio.h>

int yylex (void);

void yyerror (char const \*s);

%}

%token ONE ZERO

%%

input: {printf("Enter the line\n");}

| input line

;

line: one '\n' {printf("Correct\n");}

| zero '\n' {printf("Correct\n");}

| error '\n' {yyerrok;}

| '\n' ;

zero: zero ZERO

| ZERO;

one: one ONE

| ONE;

%%

int main() {

return(yyparse());

}

int yylex() {

int c = getchar();

switch (c) {

case '\n': case '\t': case ' ': return '\n';

case '1': return ONE;

case '0': return ZERO;

default: return c;

}

}

void yyerror(const char\* s) {

fprintf(stderr, "%s\n",s);

}

int yywrap() {

return (1);

}

# Содержимое файла спецификации c правосторонним выводом

%{

#include <stdio.h>

int yylex (void);

void yyerror (char const \*s);

%}

%token ONE ZERO

%%

input: {printf("Enter the line\n");}

| input line

;

line: one '\n' {printf("Correct\n");}

| zero '\n' {printf("Correct\n");}

| error '\n' {yyerrok;}

| '\n' ;

zero: ZERO zero

| ZERO;

one: ONE one

| ONE;

%%

int main() {

return(yyparse());

}

int yylex() {

int c = getchar();

switch (c) {

case '\n': case '\t': case ' ': return '\n';

case '1': return ONE;

case '0': return ZERO;

default: return c;

}

}

void yyerror(const char\* s) {

fprintf(stderr, "%s\n",s);

}

int yywrap() {

return (1);

}

**Список литературы**

База и Генератор Образовательных Ресурсов (bigor.bmstu.ru)

Генератор программ синтаксического анализа yacc. Производственно-внедренческий кооператив "И Н Т Е Р Ф Е Й С": Демос/P 2.1, 1988

Рейуорд-Смит В.Дж. Теория формальных языков. Вводный курс, M.: Радио и связь, 1988